TURBO: Utility-Aware Bandwidth Allocation for Cloud-Augmented Autonomous Control

Peter Schafhalter* 
University of California, Berkeley

Alexander Krentsel* 
University of California, Berkeley

Hongbo Wei 
University of California, Berkeley

Joseph E. Gonzalez 
University of California, Berkeley

Sylvia Ratnasamy 
University of California, Berkeley

Scott Shenker 
University of California, Berkeley

Ion Stoica 
University of California, Berkeley

*These authors contributed equally to this work.

Abstract

Autonomous driving system progress has been driven by improvements in machine learning (ML) models, whose computational demands now exceed what edge devices alone can provide. The cloud offers abundant compute, but the network has long been treated as an unreliable bottleneck rather than a co-equal part of the autonomous vehicle control loop. We argue that this separation is no longer tenable: safety-critical autonomy requires co-design of control, models, and network resource allocation itself.

We introduce TURBO, a cloud-augmented control framework that addresses this challenge, formulating bandwidth allocation and control pipeline configuration across both the car and cloud as a joint optimization problem. TURBO maximizes benefit to the car while guaranteeing safety in the face of highly variable network conditions. We implement TURBO and evaluate it in both simulation and real-world deployment, showing it can improve average accuracy by up to 15.6%pt over existing on-vehicle-only pipelines.

2012 ACM Subject Classification Networks → Network resources allocation; Computer systems organization → Real-time systems; Computer systems organization → Embedded and cyber-physical systems

Keywords and phrases autonomous vehicles, bandwidth allocation, cloud computing, edge computing, machine learning

Digital Object Identifier 10.4230/OASIcs.NINeS.2026.18

1 Introduction

Autonomous driving holds huge transformative potential for society, leading the first wave of real-world machine learning (ML) system applications. Autonomous vehicles (AVs) have the potential to reduce road fatalities through the elimination of human error [2], free up to one billion hours spent in traffic per day by improving traffic flow [84], and provide mobility to millions of people impacted by disabilities [22]. Recent years have seen successful limited commercial deployments of AVs [136, 24, 125] in target markets with favorable environments. However, challenges remain such as operation in poor weather conditions, construction zones, and busy regions [18].

Progress in autonomous vehicle deployment has been driven by advances in machine learning models across components of the AV control pipeline. This pipeline is structured as a directed acyclic graph (DAG) of ML-based services, each responsible for tasks such as camera stream object detection, movement prediction, and action planning (Section 2). Significant effort has been devoted to increasing the accuracy of the machine learning (ML)

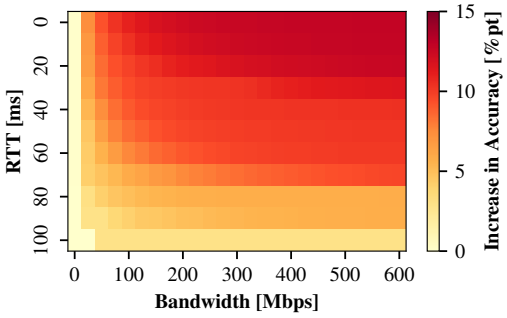


Figure 1: *TURBO improves the average accuracy of AV perception and motion prediction services across a range of network conditions.*

models [10, 139, 113, 70, 86] that implement these services, which in turn improves the end-to-end decision-making of AVs [3, 4]. Recently, state-of-the-art (SOTA) models have shown remarkable improvements in accuracy by scaling to larger parameter counts [110, 145].

However, deploying SOTA models on AVs is increasingly challenging because vehicles have an order-of-magnitude less compute than a single datacenter GPU[1]. On-car compute is fundamentally limited by physical power, thermal, and stability limits [74, 56], as well as the high cost of ML accelerators that makes scaling on-vehicle compute financially infeasible. The tight runtime service-level-objectives (SLOs) for AV control tasks – aiming to operate end-to-end with faster-than-human reaction times (*e.g.,* 0.39 to 1.2 seconds [138, 58]) – require system designers to carefully balance on-car model runtime with high-quality decision-making [44, 120, 33].

Given the growing disparity between edge and cloud compute, we ask: how can we design a real-time edge control system capable of running highly accurate, compute-intensive models without compromising safety? The key safety requirement we consider is the ability to make decisions within a real-time constraint in any operating environment. Because we aim to bridge this disparity using mobile networks, our system must continuously meet this requirement under unreliable network conditions as well as full disconnection. Under this requirement, we design our system to maximize the decision-making quality, *i.e.,* the accuracy of the individual services comprising the AV pipeline.

Meeting these design goals in a practical system presents several challenges. The primary challenge is that running the control pipeline off the car requires traversing cellular networks, which have severely limited and highly variable network conditions compared to traditional networked environments. AVs generate over 8 Gbps of data across different sensing modalities[2] [126, 47], far exceeding the 100 Mbps target uplink bandwidth for 5G networks. Prior work [106] has observed that cloud-grade GPUs can run an individual ML model faster than on-car GPUs, fast enough to make up for cellular *ping* round-trip time (RTT) latencies; however, this work does not discuss the crucial portion of latency induced by limited cellular bandwidths, which is significant enough to make executing in the cloud infeasible as shown in Section 6.1. Second, we make the key observation that the value of each data stream generated by an AV is not equal, and can vary dynamically with the environment.

[1] A single SOTA cloud GPU (H100) can perform over 10× more operations per second than AV-targeted chips like NVIDIA's DRIVE Orin [94, 92].
[2] A camera generating 1920 × 1280 frames, 10 Hz contributes 590 Mbps.

Our key insight to address these challenges is the opportunity and need to *co-design* the control pipeline with resource allocation. The opportunity arises from the fact that unlike traditional coded systems, services in a compound AI control system are *reconfigurable* through the availability of "families" of task-specific models. Each model in the family is interchangeable, providing its own runtime/size/accuracy tradeoffs for a given task. This allows our system to unify resource allocation and dynamic system configuration; *e.g.,* the system can increase bandwidth allocation to a datastream to lower transmission time and enable running a slower, more accurate model where it is holistically most beneficial. The need arises as a consequence of differing value of datastreams; as we show later on (Section 6.1), without intelligently selecting the datastreams to allocate bandwidth to, the car sees practically no performance benefit under real-world network conditions.

We present TURBO, a cross-edge-cloud AV control framework that jointly optimizes bandwidth allocation and cloud model selection to maximize benefit to the car. TURBO Task Utility and Resource Bandwidth Optimizer offers multiple cloud models for several AV services, and greedily selects the best possible configuration of both cloud models *and* bandwidth allocations that will meet strict runtime SLOs, while continuing to run on-vehicle models in parallel as a fallback to ensure baseline safety. These decisions happen dynamically at runtime, allowing TURBO to maximize overall accuracy across network conditions and driving environments. TURBO does this by extending the concept of *utility curves* [16, 130] to capture the intra-application relative benefit of a particular allocations of bandwidth across the AV system, and formulating an ILP over aggregated utility curves to select the best set of allocations.

We evaluate TURBO by running it live on a car deployment on highway and neighborhood streets, and through testing in simulation across hundreds of hours of real-world AV traces collected by Waymo [117]. Our results show that when operating with SOTA open-source models (Section 5) on a real-world AV dataset, TURBO improves average accuracy by up to 15.6 percentage points (%pt) over executing on-vehicle only and 12.7 %pt over naive bandwidth allocation methods (Section 6). Our real-world test drive shows our approach is feasible to run even with cellular network conditions today.

2 Background

2.1 Anatomy of an Autonomous Vehicle

AVs capture information about their surroundings using sensors and process that sensor data into control commands (*i.e.,* steering, acceleration, and braking). Data processing must be timely and accurate, presenting a critical challenge to the development of autonomous driving. The computation in an AV is typically structured as a pipeline where each component performs a specific task (*e.g.,* detecting nearby obstacles) [44, 126]. To ensure the timely computation of control commands, pipelines execute under an end-to-end deadline [43] with the potential to outperform human reaction times of 390 milliseconds to 1.2 seconds [138, 58].

To retrieve meaningful information from sensor data and enable intelligent decision-making, AVs employ ML models. ML models are evaluated on on large, offline datasets [75, 39, 117, 15, 7] where key innovations often result in low single-digit, but statistically significant, percentage point increases in accuracy [150, 146, 28, 80] that indicate new capabilities. Because more accurate ML models are generally more compute-intensive [110, 145, 120], constructing an AV pipeline requires careful consideration of the tradeoffs between accuracy and response time.

We identify that *AV components are services* because their tasks define concrete interfaces and their implementations select from a diverse set of potential models and algorithms. We provide an overview of the key components of an AV pipeline, examine how each component forms a service, and discuss the role of ML.

2.1.1 Sensors

AVs use high-fidelity sensors which span several modalities to observe their surroundings. *Cameras* captures images from multiple perspectives. *Lidars* generate point clouds by using a rotating array of laser beams to measure the distance to nearby obstacles. *Radars* measure the distance, direction, and velocity of nearby obstacles by emitting radio waves and measuring their reflections. *External Audio Receivers* capture sounds to detect and localize emergency vehicles sirens.

Taken together, AVs sensors capture a large amount of detailed information in order to generate a 360° view of the vehicle's environment which aims to be accurate at distances up to 500 meters [56]. To increase fidelity and provide redundancy, AVs are equipped with multiple instances of each sensor type. For example, Waymo's 5$^{\text{th}}$ generation AV uses a long-range camera to detect distant obstacles and peripheral cameras to reduce blind spots [131]. While sensor configurations vary, open-source driving datasets indicate that a single camera may generate between 479 Mbps[3] and 1.8 Gbps.[4]

2.1.2 Perception

To understand their surroundings, AVs use ML models to process sensor data into an egocentric map of nearby obstacles, driveable regions, and traffic annotations (*e.g.,* signs, traffic lights). Perception performs several different tasks such as object detection, object tracking, and lane detection [44, 9] which form subservices that may process data from different sensors. While there is a large range of perception models for autonomous driving which process different sensor modalities, most of these models use convolutional neural networks (CNNs) to extract features from images or lidar point clouds [61, 143, 121].

In this work, we examine how to allocate bandwidth across 2D object detection services, where detection on each camera stream forms a distinct service. Object detection is a well-studied perception task with a wide variety of open-source models [120, 17, 48, 12] and performs the safety-critical task of identifying and locating nearby obstacles by processing images from the AV's cameras.

Object detection models generate labeled bounding boxes that identify the positions and the classes (*i.e.,* types) of objects in an image (Figure 2), and are evaluated using the following metrics. *Average Precision* (AP) measures whether a model's predicted bounding boxes overlap with the true bounding boxes of the same class and penalizes the model for false positives. *Mean Average Precision* (mAP) reports the average AP across all object classes [111]. In this paper, we use mAP to describe the accuracy of object detection models because it is the standard detection accuracy metric across datasets and leaderboards [117, 97].

[3] Waymo provides 1920×1040 images sampled at 10 Hz [117].
[4] The Argoverse dataset [7] uses 1920×1200 cameras recording at 33 Hz.

2.1.3 Motion Prediction

Motion prediction uses ML models to anticipate the motion of nearby agents (*e.g.,* pedestrians, vehicles, bicyclists) by processing the outputs from perception. State-of-the-art prediction models typically leverage compute-intensive neural networks such as Transformers [127] to forecast the future positions of nearby agents based on patterns in their behaviors and motion [103, 112, 34]; however, some pipelines use simple compute-efficient models such as linear regression [44].

While there are several different metrics to evaluate motion prediction models, we likewise use mAP as described in [132]. For motion prediction, mAP represents agents as bounding boxes on a 2D top-down map, and uses the overlap between the predicted and true positions to estimate the accuracy at a particular point in time. When reporting the mAP, this accuracy is averaged across agent classes and time steps.

2.1.4 Remaining Components

AVs further rely on *localization* [5] to estimate the vehicle's position, *planning* to generate safe and reliable motion plans [62, 96, 52, 124, 82], and *control* to convert motion plans into steering, acceleration, and braking commands [45]. While localization provides motion prediction with access to high-fidelity maps, planning and control are downstream components that rely on accurate results from perception and motion prediction to make driving decisions.

2.2 Remote Interventions

AVs already rely on cellular networks for safety-critical decision-making for remote interventions. When in an uncertain situation (*e.g.,* construction zones), the AV contacts a remote human operator for guidance [83, 123]. Using transmitted sensor feeds and annotated representations of the AV's surroundings, the operator makes informed decisions to help the AV proceed *e.g.,* by creating a route for the AV to follow or answering clarifying questions posed by the AV.

3 Motivation

Highly accurate ML models are key to ensuring that AVs can make safe and reliable decisions. These models enable processing huge volumes of sensor data into a detailed representation of an AV's surroundings, as well as understanding and forecasting the behaviors of nearby human agents. However, technical and economic trends challenge the ability to deploy the highest-accuracy models.

The scalable ML model architectures [110, 145, 80] (*e.g.,* Transformers [127]) that underpin state-of-the-art models for tasks pertinent to driving applications [71, 17, 90] have also increased the computational resource requirements for inference by greatly increasing the total number of model parameters. For this reason, ML practitioners develop *families* of models with different tradeoffs between the number of parameters and the accuracy of a model (Figure 2). For example, the EfficientDet family of object detection models [120] uses a scalable convolutional neural network architecture [119] to develop 9 different models with varying parameter counts and accuracies. The smallest model, EfficientDet-D0 contains 3.9M parameters, requires 2.5B floating point operations (FLOPs) to process a single image, and achieves a mean average precision (mAP) of 34.3% on the COCO validation dataset [75].

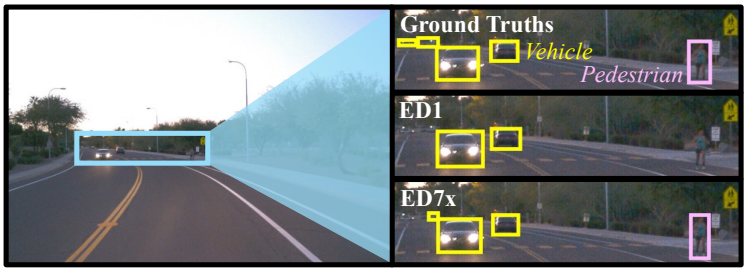


Figure 2: ***More accurate detectors provide a better understanding of the surroundings.*** *In this scene from the Waymo Open Dataset [117], EfficientDet-D1 (ED1 – center) detects the two nearby vehicles but misses the pedestrian on the side of the road, resulting in a mean average precision (mAP) of 25%. In contrast, highly accurate ED7x model (bottom) detects the pedestrian and both nearby vehicles, but generates an incorrect bounding box for one of the distant vehicles, resulting in an mAP of 75%.*

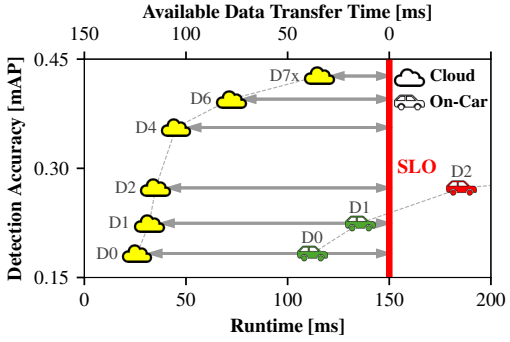


Figure 3: *Cloud accelerators can execute more accurate models with lower runtime than hardware designed for autonomous driving, enabling AVs to increase accuracy while meeting stringent SLOs.*

In contrast, the largest model, EfficientDet-D7x, contains 77M parameters, requires 410B FLOPs to process a single image, and achieves a mAP of 54.4%.

Unfortunately, on-vehicle compute is limited by a variety of technical and economic constraints. Prior work [74] shows AV compute can significantly degrade the driving range of electric AVs due to the energy needed for computation and cooling, and recommends that AV hardware must tolerate significant impulses and vibrations. Moreover, the cost of deploying SOTA ML accelerators on AVs is prohibitive; an NVIDIA H100 GPU costs $30k-40k in 2024 [115, 114], as much as a new Tesla Model 3 [13], and cost is already a key factor in the design of AVs [56]. Along with spatial constraints, these limitations prevent AVs from deploying powerful chips or scaling compute on-vehicle.

Therefore, AV manufacturers leverage cost-effective compute platforms such as NVIDIA's DRIVE Orin which powers autonomous driving for Volvo and SAIC [68], but performs over 10× fewer operations per second than the H100 [94, 92]. Due to these hardware limitations, AVs lack the processing power to execute the most accurate models in real time. Thus, AV developers must carefully navigate the tradeoffs between runtime and accuracy at design time (*e.g.,* by selecting appropriate models and using techniques such as quantization) to ensure that AVs can make high-quality and safe driving decisions while providing rapid response times [44, 43].

Prior work in robotics observes that the cloud model runtime is fast enough to enable

running more-accurate models within-SLO, while continuing to run an on-vehicle model as backup [106]. However, their work does not focus on the effect of limited *bandwidth*, only accounting for network latency. As we show in (Section 6.1), ignoring bandwidth limits accuracy improvement to under 2%, as model inputs are unable to reach the cloud in time to finish model execution by the SLO. The focus of this paper is on the impact of the network and specifically the role of intelligent bandwidth allocation on real-time edge control system accuracy. We show that careful bandwidth allocation is crucial, unlocking up to 6× higher accuracy improvement. In addition, we present a complete end-to-end system implementation and evaluation.

4 Method

The goal of our method is to maximize benefit to the car's safety by choosing which services to run in the cloud. In order to do this, we must decide how much bandwidth to allocate to each service based on currently available bandwidth and ping latency, then use these settings to execute the best-performing model which satisfies the service-level objectives (SLOs). To achieve this, we turn to the idea of "utility" as introduced in previous work [16], which defined utility as the incremental benefit an application gets from an additional unit of bandwidth, and observed that real-world applications may have highly-variable utility for each incremental unit of bandwidth.

We observe that the compound AI system structure [107] of the AV control pipeline introduced in Section 2 decomposes cleanly into individual services that have their own distinct "utility curves" mapping bandwidth allocation to overall control accuracy impact. Thus our method has two parts; first, at system design time, we compute the utility of each available model as a function of allocated bandwidth (Section 4.1), then compose the utility functions of all available models to generate the utility functions of each service (Section 4.2). Second, at runtime, using these utility functions, we formulate and solve an Integer Linear Program (ILP) to find the optimal bandwidth allocations for the ping latency and amount of bandwidth available (Section 4.4) that maximizes the application-level utility (Section 4.3). We describe each part below.

4.1 Model-Level Utility

A model's utility must take into account the model's performance on its task (*i.e., accuracy*) as well as whether the model meets its SLO (*i.e., latency*). Given a model m with accuracy A_m which must execute within a latency SLO t_{SLO}, we design the utility as a unit step function which provides a utility of A_m if t_{SLO} is met. If t_{SLO} is missed, we return a utility of 0 because AVs must make decisions in real-time and late results provide no value. We observe that allocated bandwidth b and round-trip time (RTT) t_{RTT} affect the observed runtime of a model $T(b, t_{\text{RTT}})$, leading to the utility function:

$$U_m(b) = \begin{cases} A_m & T(b, t_{\text{RTT}}) \leq t_{\text{SLO}} \\ 0 & T(b, t_{\text{RTT}}) > t_{\text{SLO}} \end{cases} \tag{1}$$

Because on-vehicle models are configured to meet latency SLOs and are unaffected by network transfer times, the utility curves of on-vehicle models simplify to:

$$U_{\text{on-vehicle model}}(b) = A_{\text{on-vehicle model}} \tag{2}$$

In contrast, the runtime of models executing on remote resources depends on the allocated bandwidth, the characteristics of the model, and the network conditions. Given model input size S_{input}, round trip time t_{RTT}, and execution time t_{exec}, the total runtime of the model is:

$$T_{\text{remote model}}(b, t_{\text{RTT}}) = t_{\text{exec}} + t_{\text{RTT}} + \frac{S_{\text{input}}}{b} \tag{3}$$

This produces the following utility curve which is a step function from 0 to $A_{\text{remote model}}$ where the step occurs at b_c when $T(b_c, t_{\text{RTT}}) = t_{\text{SLO}}$:

$$U_{\text{remote model}}(b) = \begin{cases} A_{\text{remote model}} & t_{\text{exec}} + t_{\text{RTT}} + \frac{S_{\text{input}}}{b} \leq t_{\text{SLO}} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$b_c = \frac{S_{\text{input}}}{t_{\text{SLO}} - t_{\text{exec}} - t_{\text{RTT}}} \tag{5}$$

A key consequence of the step-utility function is that b_c is the minimal amount of bandwidth to run a remote model which is optimal when bandwidth is scarce. Bandwidth allocations greater than b_c provide the same utility of A_m as a bandwidth allocation of exactly b_c; consequentially, any excess bandwidth is wasted. Moreover, any bandwidth allocation less than b_c has a utility of 0, so bandwidth allocations $0 < b < b_c$ are also wasted.

We also observe that more accurate models typically have larger inputs S_{input} and longer execution times t_{exec}. Consequently, more accurate models typically require larger bandwidth allocations b and are less tolerant of large round trip times t_{RTT} compared to their less accurate counterparts.

4.2 Service-Level Utility

A *service* consists of a specific task, such as detecting obstacles on a particular camera stream, as well as several models that can perform that task with different parameters and resource requirements. The service typically runs on-vehicle the best model that on-vehicle resources allow, and additional models can execute remotely on cloud-based hardware. To ensure that the service tolerates sudden disconnections and reductions in bandwidth, the service always executes the on-vehicle model as suggested in [106]. In this way, **we guarantee that the service will always provide a utility of at least $A_{\text{local model}}$**, ensuring that our approach of attempting to use remote models is *strictly better* than using only local models.

We define the "utility function" of a service s as the amount of control-application utility gained by granting a certain amount of bandwidth b to the service.

To ensure the service selects the most accurate model that satisfies the SLO, we define the service-level utility as the maximum utility among all of the service's models $\mathcal{M}$:

$$U_s(b) = \max_{m \in \mathcal{M}} U_m(b) \tag{6}$$

By applying the model utility function defined by Equation (1), we find that $U_s(b)$ equals the utility of the most accurate model that can satisfy its SLO with the provided bandwidth.

We show an example of the construction of a service-level utility curve for a single object detection service which processes a video stream using a small EfficientDet D1 (ED1) model available on the car and two larger models, ED3 and ED5, available in the cloud, shown in

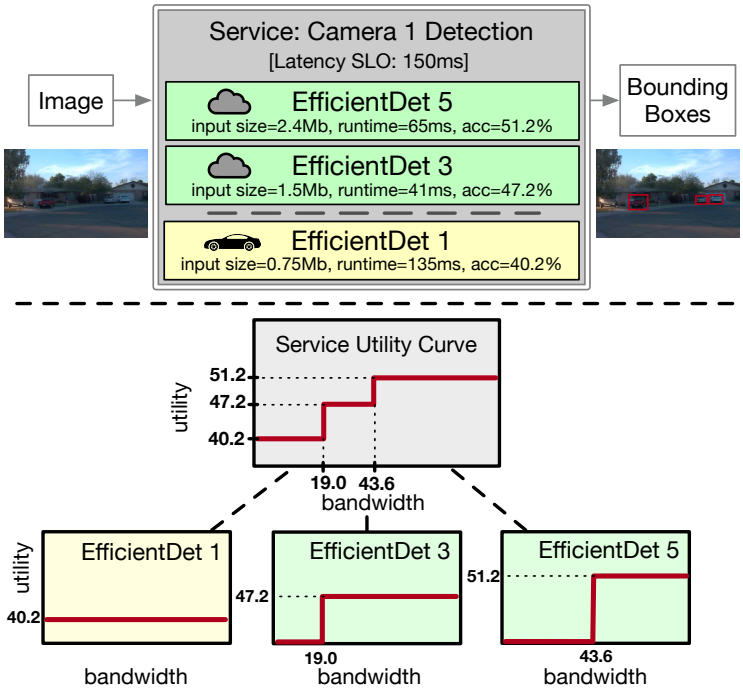


Figure 4: *Utility curves for a service with 3 object detection models.*

Figure 4. Because more accurate models require more bandwidth to meet their SLOs, the shape of $U_s(b)$ is a series of steps which occur wherever the service has sufficient bandwidth to transition to a more accurate model. Thus, the optimal bandwidth allocation for a service is the selected model's optimal bandwidth b_c, or 0 if running a local model, and any excess bandwidth is wasted.

Under our definition, each service executes at most 1 remote model[5]. Therefore, setting a bandwidth allocation for a service effectively selects which model (or none) to run remotely.

4.3 Application-Level Utility

To achieve optimal performance on application level goals (*e.g.,* safety for autonomous driving), we must coordinate bandwidth allocation across services. The key challenge lies in the fact that the relationship between service-level utility and AV control system quality is not always well-defined and may vary across different services; that is, overall service performance is not necessarily the average accuracy across all services, but rather the safety of the final end-result of how the car moves.

For example, detecting obstacles observed by the AV's front camera is typically more important than detecting obstacles from the AV's side camera. Complicating the matter, the performance of one service may impact the performance of another given the DAG structure of compound AI systems, *e.g.,* motion prediction relies on an accurate history of the motion of nearby obstacles, and the accuracy of this history is impacted by the detection accuracy.

[5] It may also be possible to run multiple cloud models and return results from each as they become available using the best results returned before the SLO, however this may become computationally cost prohibitive; we do not consider such a design here.

We propose a general framework which derives an overall application-level utility by combining the utility functions of a set of services $\mathcal{S}$:

$$U_{\text{app}}(b) = \max_{b_s : s \in \mathcal{S}} \sum_{s \in \mathcal{S}} f_s\big(U_s(b_s)\big) : \sum_{b_s : s \in \mathcal{S}} b_s \leq b \tag{7}$$

We account for differences in the impact of each service's utility on the application-level utility by transforming the service-level utility with $f_s : \mathbb{R} \to \mathbb{R}$. For example, setting $f_s(x) = ax + b$ can re-weight service-level utilities to prioritize more important services. The f_s transformation may also be used to normalize the utilities of different services so that they exist in the same range and can be more easily compared (e.g., by incorporating a sigmoid function to convert $U_s : \mathbb{R}^+ \to \mathbb{R}$ to $f_s \circ U_s : \mathbb{R} \to [0, 1]$. We emphasize that f_s should be chosen carefully, and evaluated or even learned using real-world data or highly realistic simulations to ensure that U_{app} reflects the application's goals.

4.4 Runtime Bandwidth Allocation

Our goal at runtime is to decide which models to run in the cloud and thus which input data to transfer to the cloud with the limited bandwidth available. In order to do this, we collect periodic estimates of available bandwidth and RTT using standard methods [32, 36].

We formulate the bandwidth allocation decision as a utility maximization problem, given the available bandwidth, RTT, and the utility functions for the models (Section 4.1), services (Section 4.2), and application (Section 4.3). Our formulation is an Integer Linear Program (ILP) that maximizes the utility of the overall application, and leverages the fact that the utility functions of models and services are step functions directly select which configurations of cloud models to run.

Let $|\mathcal{S}|$ be the number of services, and note that steps of the utility function for each service $s \in \mathcal{S}$ can be defined by the accuracy $a_{s,m}$ and the location of the step $b_{c,s,m}$ for each model configuration $m \in \mathcal{M}_s$ in the service.

We define binary decision variables:

$$x_{s,m} = \begin{cases} 1 & \text{if model } m \text{ of service } s \text{ is selected,} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Objective Function: Maximize the total utility across all services:

$$\max_{x} \quad \sum_{s \in \mathcal{S}} \sum_{m \in \mathcal{M}_s} x_{s,m} \cdot a_{s,m} \tag{9}$$

Constraints: Ensure that cumulative bandwidth allocated is below the available total, and that only one or zero cloud-based model configurations are selected.

Bandwidth Constraint: The total allocated bandwidth across all services must not exceed the available bandwidth:

$$\sum_{s \in \mathcal{S}} \sum_{m \in \mathcal{M}_s} x_{s,m} \cdot b_{c,s,m} \leq B \tag{10}$$

where $b_{c,s,m}$ is the bandwidth at which the step of the utility function for model m of service s occurs.

One Cloud Model per Service Constraint: Each service must be allocated one model configuration:

$$\sum_{m \in \mathcal{M}_s} x_{s,m} = 1 \quad \text{for each } s \in \mathcal{S} \tag{11}$$

Binary Constraint: The decision variables are binary:

$$x_{m,s} \in \{0, 1\} \quad \text{for all } m \in \mathcal{M}_s, s \in \mathcal{S} \tag{12}$$

The objective function maximizes the sum of utilities across all selected intervals. The bandwidth constraint in Equation (10) ensures that the total allocated bandwidth does not exceed the available bandwidth. The one model per service constraint in Equation (11) enforces that each service selects one model configuration, which indicates that the solver should not consider solutions that run multiple models for a service in the cloud and waste bandwidth. The decision variables are constrained to be binary, reflecting the discrete nature of the utility allocation.

This ILP formulation directly selects which models configurations to run for the cloud across a set of services while maximizing the overall utility to the application. Consequentially, solving the ILP also generates bandwidth allocations for each service that are defined by the step b_c of the selected models' utility functions. In Section 5, we apply this method to derive utility functions using state-of-the-art models, and evaluate its ability to boost the accuracy of autonomous driving services in Section 6.

5 Design and Implementation

We design and implement the method described as a standalone control module. To evaluate its efficacy in a real-world environment, we additionally design and implement a edge-cloud offload control system runtime, which we evaluate in Section 6.2. We describe the design for the module and the wider system in turn below.

5.1 Resource Model

We model the AV as a resource-constrained edge device with an NVIDIA Jetson Orin compute system because it uses the same chip as the NVIDIA DRIVE Orin [93] which powers autonomous driving for vehicles from Volvo and SAIC [68]. We assume that the AV has sufficient compute resources to run a pipelined AV software stack using resource-efficient ML models on-vehicle that meet a minimum accuracy level and runtime SLO. We view the cloud as a resource-rich compute environment which can immediately process incoming data with more accurate ML models. For cloud hardware, we assume that models execute on an NVIDIA H100 GPU from a RunPod 1×H100 PCIe instance with 176 GB RAM and 16 vCPUs. While there is significant work on serving systems for ML models, many take advantage of economies of scale and batching to improve efficiency [23, 104, 46]. We consider these systems to be complementary to our work as they improve resource utilization and reduce costs when serving models at scale.

Model	Input [Mb]	Preprocessing [ms]		Inference [ms]	
		Orin	H100	Orin	H100
ED1	9.8	18	11	118	21
ED2	14.2	20	12	166	23
ED4	25.2	25	15	523	30
ED6	39.3	37	18	1350	54
ED7x	56.6	43	25	2320	91

Table 1: *EfficientDet models. Preprocessing measures the runtime of resizing and preparing the image on CPU. Inference includes transferring pre-processed data to GPU and running the model.*

5.2 Tasks

5.2.1 Object Detection

We first examine how object detection, a representative computer vision task, can benefit from cloud computing. Object detection is a well-studied task with a wide range of open-source models [12, 102, 120, 17] and datasets [26, 75, 117, 39, 144] and uses model design patterns (*e.g.,* convolutional neural networks) which are common in other perception tasks including semantic segmentation and 3D object detection. Our object detection SLO is 150 ms[6] which is similar to the perception runtimes of existing systems [43, 6].

Models. We study the EfficientDet family of models [120] because they provide a large trade-off space between latency, accuracy, and resource requirements (Table 1). We select **E**fficientDet-**D1** (ED1) as our on-vehicle model because ED1 is the most accurate model that can meet the SLO using on-vehicle hardware (Figure 3). Our cloud models are ED2, ED4, ED6, and ED7x, which are increasingly accurate and resource-intensive and are all unable to meet the SLO using on-vehicle hardware. EfficientDet models include a lightweight on-CPU preprocessing step to resize and prepare images for the deep neural network (DNN) running on the GPU. We exploit this preprocessing step as well as compression on images and the along to generate the following configurations for EfficientDet inference with different tradeoffs between runtime, accuracy, and the amount of data to transfer (Figure 5):

1. *Cloud preprocessing* transfers the original image to the cloud for preprocessing and neural network execution. This configuration shifts all processing to the cloud, resulting in the fastest runtime but the largest data transfer size.
2. *On-vehicle preprocessing* preprocesses the image on-vehicle and transfers the smaller inputs to the cloud. This configuration trades slower on-vehicle preprocessing for smaller data transfer.
3. *Image compression* compresses the original image using lossless PNG or lossy JPEG compression and transmits the compressed image to the cloud, where it is then decompressed. This configuration further reduces the amount of data transferred at the cost of higher runtimes due to compression and decompression.
4. *DNN input compression* preprocesses the image on-vehicle and similarly applies lossless PNG or lossy JPEG compression to the preprocessed model inputs which are then transmitted to the cloud. This configuration has the largest reduction in data size but the

[6] Object detection models may meet tighter SLOs with model compilers, specialized runtimes, and model compression techniques. We emphasize that our goal is to examine a representative task for perception with a representative SLO in which cloud computing permits the execution of more accurate and resource-intensive models.

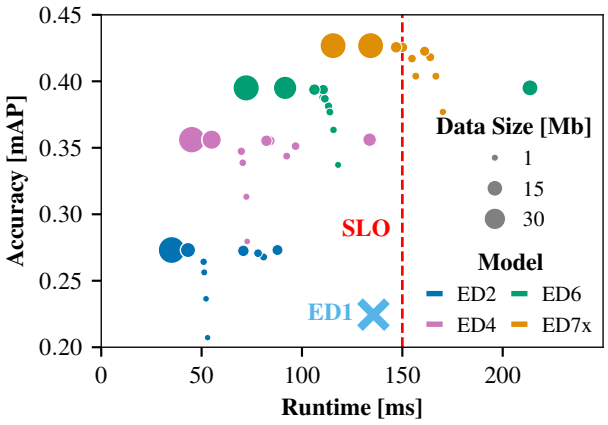


Figure 5: *We consider 35 model configurations with different tradeoffs in runtime, accuracy, and data transfer size: ED1 runs on-vehicle while ED2, ED4, ED6, and ED7x execute in the cloud. We generate additional configurations by applying lossless PNG and lossy JPEG compression to either the original image or the pre-processed model inputs. Our JPEG quality factors are 95, 90, 75, and 50. We find that compression significantly reduces the amount of data to transfer at the cost of increased runtime and reduced accuracy.*

slowest (local) runtime.

Dataset. To train and evaluate detection models, we use the Waymo Open Dataset (WOD) v2.0.0 [117] which consists 6.4 hours of driving organized into 1150 scenes of 20 seconds sampled at 10 Hz and contains 9.9 million annotated bounding boxes. The WOD provides 5 camera perspectives which we view as distinct tasks. The front, front-left, and front-right cameras generate images at resolution of 1920×1280, while images from the side and rear cameras are 1920×886, resulting in uncompressed data sizes of 59 Mb and 41 Mb respectively. The P99 number of ground truth bounding boxes per image is 62, translating into an output size of 7.9 Kb.[7]

Training. We partition the scenes from the WOD v2.0.0 [117] into training (68%), validation (12%), and test (20%) sets. To adapt the models to the dataset, we modify their classification heads to recognize five classes of objects: vehicles, pedestrians, cyclists, signs, and other. We initialized the models with pre-trained weights from the COCO dataset [75] provided by [137], and fine-tune on the WOD dataset for 10 epochs[8].

5.3 Motion Prediction

Motion prediction is a critical autonomous driving task which estimates the future positions of nearby agents (*e.g.,* vehicles, pedestrians, and cyclists). Motion prediction is an active area of research with a variety of approaches using different neural network architectures. We select an SLO of 250 ms for motion prediction based on the reported runtimes of open-source AV implementations [43, 6]. While the motion prediction models take both high-definition (HD) maps and historical agent trajectories as inputs, we only transmit agent information as maps can be pre-computed and stored in the cloud.

[7] Each bounding box comprises four 32-bit floating point numbers for the minimum and maximum x and y values, and an 8-bit integer for the class.
[8] We fine-tune ED7x for only 8 epochs due to the high cost of training.

Models. For the cloud model, we use Motion Transformer [112], a state-of-the-art model that employs a Transformer architecture to forecast trajectories and ranks first on the 2022 Waymo Open Motion Dataset[9] [33] (WOMD) leaderboard. For on-vehicle predictions, we select MotionCNN [64], a lightweight model that ranked third on the 2021 WOMD leaderboard. MotionCNN represents trajectories and surroundings as a fixed resolution image and uses a convolutional neural network to predict future paths.

Dataset. To train and evaluate the motion prediction models, we use WOMD v1.2.1 which contains over 100k scenes of 20 seconds sampled at 10 Hz split into a training (70%), validation (15%), and test (15%) sets. The WOMD includes 3D bounding boxes for each agent and map data (*e.g.,* lanes, signs, crosswalks) for each scene. Models must predict the positions of selected agents at three, five, and eight seconds in the future using an HD map and a one second trajectory history of each nearby agent. Thus, the Motion Transformer model running in the cloud has a P99 input size of 146 Kb and a P99 output size of 246 Kb.

Training. We modify the open-source implementations of Motion Transformer and MotionCNN for compatibility with WOMD v1.2.1, and follow the published training procedures. Our trained models report a validation accuracy of 0.22 mAP for MotionCNN and 0.40 for Motion Transformer.

5.4 Utility Functions

We follow Section 4.1 to design utility functions for each model configuration. Our SLOs are 150 ms for object detection and 250 ms for motion prediction. We profile each cloud model using the Jetson Orin for pre-processing and compression and the H100 for pre-processing, inference, and post-processing and compute t_{exec} using the P99 values. We also measure the amount of data to transfer across the network for each inference iteration and likewise set S_{input} to the P99 value. While images and pre-processed EfficientDet model inputs have constant sizes, their compressed sizes vary (Figure 5). Likewise, the Motion Transformer's input size varies with the number of agents in the scene[10]. Based on these profiles, we use Equation (5) to calculate the bandwidth at which the step occurs b_c which, along with the accuracy, characterizes the model configuration's utility function.

To calculate the service-level utility, we include all cloud models as well as the on-vehicle model. Because the on-vehicle model does not transmit data, it provides a floor to the performance of the service. We further set $f_s(u) = u$ when calculating the application-level utility. Therefore, the application-level utility equals the average accuracy across all services.

5.5 Control Module Implementation

We model our ILP in Python using PuLP [105] using the measured accuracy, runtime, and data transfer requirements measured in Section 5.2 in conjunction with the RTT and available bandwidth according to Section 4.4. We use the CBC solver [38] to solve the ILP which selects the cloud model configurations the object detection and motion planning services.

[9] Waymo's Open Dataset and and Open Motion Dataset are distinct
[10] We only transmit agent information because map data can be pre-computed and stored.

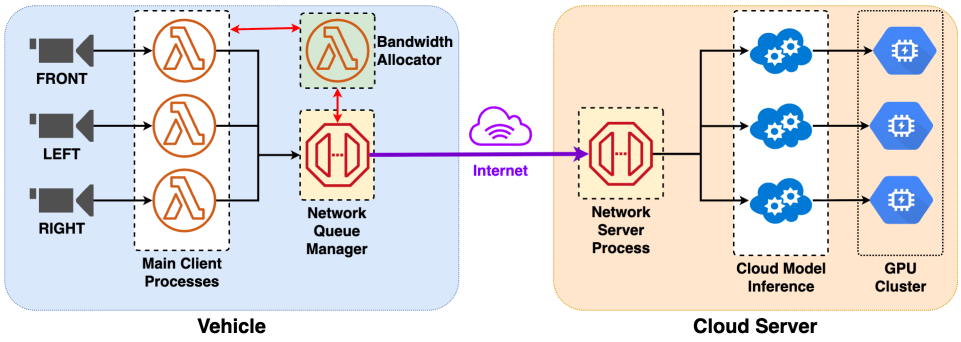


Figure 6: *A high-level overview of the system implementation.*

5.6 Edge-Cloud Offload Runtime Implementation

We demonstrate the real-world feasibility of AV bandwidth allocation by providing an end-to-end system implementation involving real-time network monitoring, edge cloud offloading, bandwidth allocation/multiplexing, and live camera feed ingestion. We design our system around the following tenets: to 1) minimize the latency incurred throughout the system; 2) ensure that a local on-car inference result is always available as a fallback if an offload request to the edge-cloud times out of the runtime SLO; and, 3) ensure that the bandwidth control module uses real-time monitors of network RTT and bandwidth conditions.

At a high level, our system implementation (Figure 6) consists of two main parts: an on-vehicle client and the cloud server. The on-vehicle portion consists of a collection of on-car sensors (e.g. cameras); a main client process that performs preprocessing and compression on the read camera images and writes them to the network queue process; a network queue manager that transmits images, receives cloud responses, enforces bandwidth constraints via queuing for each of the control services, and monitors network CWND and RTT conditions; and, a bandwidth allocator that runs our ILP optimization formulation based on the monitored network conditions. The cloud server portion consists of a network server process that receives and transmits from the on-vehicle network process, and PyTorch processes that run preprocessing (if necessary) and inference on received camera images. The system totals 1570 lines of Rust, and 3052 lines of Python.

For the network process that manages transfers between the on-vehicle and server portions of the system, we elect to use `s2n-quic`, a highly-optimized industry implementation of the QUIC protocol [109]. We choose to use a QUIC implementation as our transport medium because the QUIC protocol [55] has two main features essential in our application: it (1) supports concurrent traffic streams on the same QUIC connection without head-of-line blocking, and (2) provides key live metrics about network conditions such as the RTT and CWND, which are necessary inputs to our bandwidth allocator control module. These features are enabled by QUIC running directly over UDP and bypassing the OS TCP stack, allowing for greater transparency about network conditions and fine grained control over traffic streams.

6 Evaluation

We seek to answer two key questions about our design; (1) how much benefit in accuracy does the AV receive from using TURBO and how is this accuracy impacted by the specific design choices we make, and (2) is our approach technically and economically feasible?

We evaluate in two stages. First, in Section 6.1 we explore our system performance on

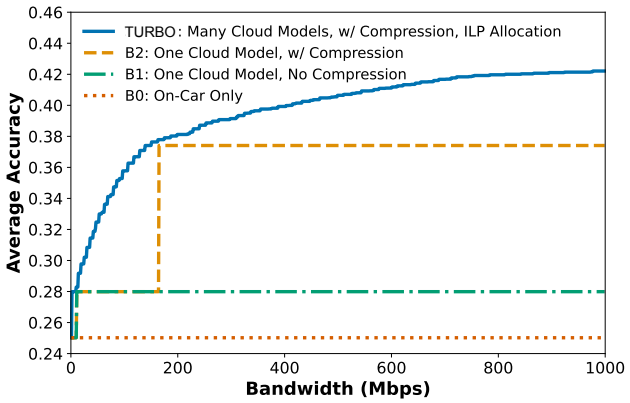


Figure 7: *Average accuracy across services for TURBO compared to baselines of varying naivete as bandwidth increases. We assume an RTT of 20ms, object detection SLOs of 150ms, and motion planning SLO of 250ms, averaging performance across all scenarios.*

a range of network conditions in simulation using thousands of real-world production AV traces from the Waymo Open Dataset [117] described in Section 5.2. As the scenarios are pre-recorded real-world data, we cannot modify the car's action during its drive, or detect a change in frequency of "disengagements" [95] or crashes [134, 122] as reported in safety cards by real-world AV fleet operators. However, we are able to quantify exactly how much service accuracy increases on the real-world sensor data, which leads directly to improved environment perception and thus more faithful planning. We show in Section 3 examples from the Waymo dataset where better models are able to detect a mid-distance pedestrian that the on-car model simply misses. Second, in Section 6.2 we outfit a car with cameras, a mobile hotspot, a local server, and a cloud-hosted server, and run TURBO during a testdrive, reporting the system's real-world performance.

In all settings, we select the identity function $f_s(x) = x$ as our "re-weighting function" for all services (Section 4.3) *i.e.,* maximizing average accuracy across services, however we note that alternative prioritization policies are possible.

6.1 Simulation Performance

Our simulation control pipeline contains six services (as described in Section 5.2); five detection services running for each of the five camera views available in the dataset, as well as the motion prediction service.[11] We evaluate on-car and cloud performance on the hardware reported in Section 5.1, measuring the pre-processing, compression, decompression, and model runtime for each model configuration on both the car and cloud hardware, then constructing the end-to-end runtime of each selected configuration across car and cloud.

TURBO Accuracy vs. Baselines. Figure 7 presents our method's performance compared to three tiers of baselines (B) of varying naivete as we vary the available network bandwidth, with a fixed RTT of 20ms (per [88] measured for a server within 500km). B0 shows accuracy

[11] Motion planning depends on results of object detection; in this work, we assume pipelined execution of the AV control program [44, 126, 8], thus bundle motion planning using the results of earlier perception along with perception on newly collected data. While it is conceivable to optimize further by moving multiple sequential dependencies to the cloud to avoid round-trips to the car, it is out of scope for our work; we leave such an investigation to future work.

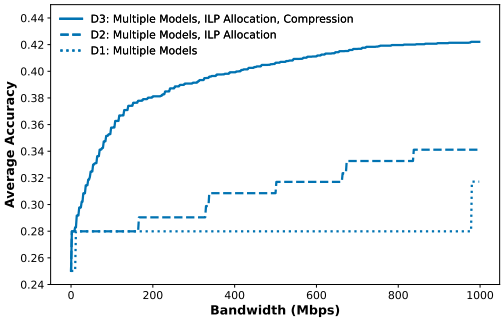


Figure 8: *Incremental benefit from TURBO's design decisions. D1 naively uses multiple models to increase accuracy. D2 uses an ILP to effectively distribute bandwidth to services. D3 uses compression to reduce data transmission size.*

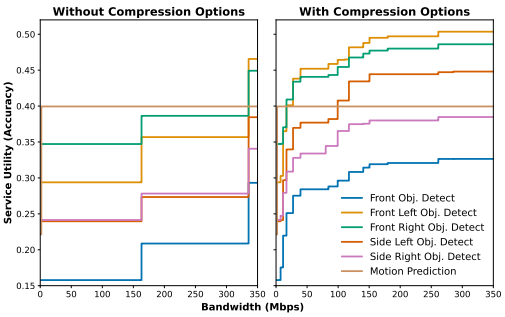


Figure 9: *Utility curves for each service as derived by TURBO, without (left) and with (right) compression options considered, showing each service's accuracy if allocated a certain amount of bandwidth.*

when running only on the car as is standard today, using the highest-accuracy model for each service that can run within the service SLO (ED1 for obj. detect per camera, a CNN-based model for motion planning). B1 shows the accuracy resulting from offering one cloud model for each service (ED4 for obj. detect per camera, a transformer-based model for motion planning) in addition to the on-car models running as backup, as proposed in prior work [106]. Finally, B2 applies object detection input preprocessing and compression (ED4 with preprocessing on-car, and compressed JPEG at 90% quality) on the car before transferring over the network, still without special consideration to bandwidth allocation across services.

TURBO (solid blue) improves on these baselines by up to 15.6%pt by being bandwidth and utility-aware; this allows us to both allocate bandwidth optimally, and select the right preprocessing and compression to run for our model choice, based on the available bandwidth. We observe that in B1, after the initial bump due to the small-input-size motion planning service being able to move to the cloud, there is no benefit to any of the camera services until well over 2Gbps (not shown) as without special attention to bandwidth allocation as we propose, bandwidth is split equally amongst the services by TCP [21]. B2 simply shifts this big step up earlier without fixing the underlying issue, and still provides only a single "step" point. In contrast, TURBO is able to quickly allocate bandwidth to the services that see the largest benefit (such as motion planning, corresponding to the jump at the left of the graph, as its inputs are relatively small and accuracy gains relatively big per Section 5.3), then continually allocate bandwidth as necessary to achieve maximal accuracy.

Factor Analysis TURBO makes three key design ideas to get its improved performance: (D1) adding multiple models to select from based on the available bandwidth, (D2) formulating that selection as a utility maximization ILP, and (D3) dynamically applying one of a range of different compression configurations. Figure 8 shows the relative contribution of each.

TURBO's gains are attributed to the synthesis of the three; providing multiple cloud models of varying sizes per service allows services an earlier feasible step up to better models (980Mbps in Figure 8), however that happens in lock-step as bandwidth is shared equally across services by default. TURBO's ILP utility maximization formulation for bandwidth allocation allows individual services to "upgrade" cloud models in an order that it optimal

for the AV's overall utility. Finally, adding compression provides many more opportunities at much lower bandwidths to step up to better cloud models, which provides the ILP formulation with many more points on which to optimize, as shown in Figure 9.

Varying Network Conditions TURBO provides variable benefit based on the network conditions available. Figure 1 shows the mean increase in accuracy across all scenarios for a range of bandwidth and RTT network conditions. While higher bandwidth and lower latency always provides more benefit, we see up to 10%pt higher accuracy with as little as 150Mbps with RTTs of 20ms, well within the experienced operating ranges provided by 5G [54, 88, 41].

6.1.1 Dynamic Utility

In performing our analysis, we observed that a model's true accuracy can vary quite widely across frames, and further that these distributions vary greatly across services and camera angles (Figure 10). To this end, we explored adapting our method using *dynamic* utility curves rather than using the average accuracy calculated from an offline dataset. Our dynamic utility curves are derived from a recent group of recent frames in order to adapt to the current driving environment. We evaluate four different policies for generating dynamic utility curves with varying degrees of freshness:

1. **Global Static**: static utility curves derived from average accuracy across all frames of all scenarios, used above.
2. **Scenario Static**: utility curves static to each scenario in the dataset, derived from the average accuracy across all frames of the particular scenario.
3. **N-Windowed**: utility curve computed from the N^{th} frame of each scenario, used for the next $N-1$ frames. We experiment with $N = 10, 20, 30, 50$.
4. **Per-Frame Oracle**: optimal ground-truth utility curve for each frame for each scenario, using the actual accuracy of each model on that frame.

Figure 11 shows the distribution of accuracy improvement over using on-car only models. The Per-Frame Oracle policy serves as a performance upper-bound, showing the best-possible accuracies one could achieve with full information: an upper bound of +11.30%pt median improvement over the on-vehicle models and +1.84%pt over the global static policy.

However, recomputing utility curves for every frame is not practical. A more practical dynamic policy such as the windowed N=20 policy, provides a 10.51%pt median improvement over the on-vehicle models and +1.05%pt over global static. Thus, we conclude that dynamically adjusting utility provides a small but meaningful increase in accuracy over our static policy, indicating that AVs can benefit from adjusting bandwidth allocations to account for changes in model accuracy when driving in different environments.

Accurate estimation of ground truth. While the "global static" curve can be computed on a large collection of previously collected data, dynamic curves require some way of estimating model accuracy in a given environment and window. Finding methods for doing so accurately is an open problem [108], and thus beyond scope of the work. We observe that our requirement is weaker and that we can simply determine relative accuracy difference between the models. To this end, we propose one way may be to periodically upload high-resolution images on spare bandwidth to run the best model and all models, and estimate relative accuracy by the magnitude of difference from a state-of-the-art model. Evaluating strategies for relative accuracy estimation is out of scope for this work. For our system implementation and real-world performance testing, we used the pre-computed global static policy.

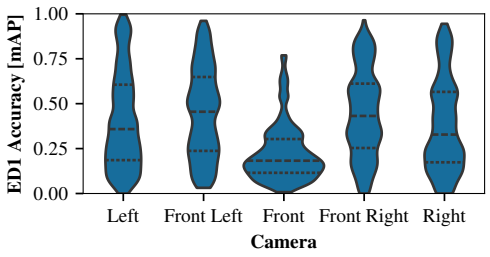


Figure 10: *Distribution of accuracy across all frames for each camera service when using the ED1 model. The accuracy distributions vary greatly, as the distribution of object types and scene properties differ based on positioning.*

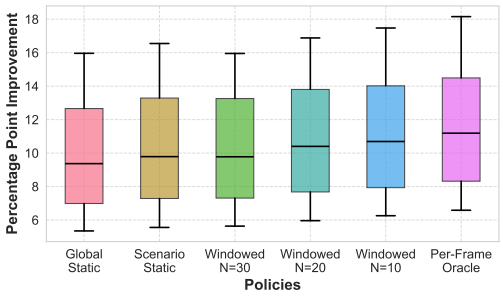


Figure 11: *Distribution of %pt improvement over on-car-only models for each policy, ranging from most static on the left to most dynamic on the right, assuming 250Mbps of bandwidth and 20ms RTT. 5, 25, 50, 75, 95th percentiles shown.*

(a) *External view of the test car's window-mounted antenna.*

(b) *Interior view showing the mounted camera and local laptop server.*

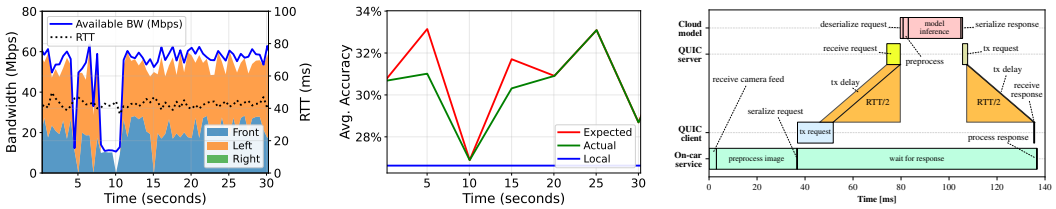
Figure 12: *Experimental vehicle setup.*

6.2 Real-World Performance

We run TURBO on an actual vehicle under real-world operating conditions, aiming to answer (1) what are TURBO's performance characteristics in the face of fluctuating network conditions and (2) how well actual accuracy benefit matches the system's expected benefit?

Experiment Setup. We outfitted our test car, a 2024 Ford Explorer, with the following hardware (shown in Figure 12); 3x Logitech Brio 4k webcams capturing front, right, and left videos streams; a consumer-grade NETGEAR NightHawk M6 Pro Mobile Hotspot with a T-Mobile SIM card; an omnidirectional window-mounted MIMO cellular antenna; an ROG Zephyrus G14 laptop with 8-core 4GHz processor and an NVIDIA GeForce RTX 4060 GPU, with direct ethernet connection to the hotspot; a 300W car plug power inverter to power the laptop and hotspot. For our remote server, we reserved an H100 GPU on Google Cloud; as nearby GPUs were unavailable at the time of our experiment, we reserved the closest available GPU 600 miles away.

We drove for 2 hours in a medium-sized metropolitan area in the United States, alternating between freeways and neighborhoods. Our car control system was responsible for ingesting and processing the data streams coming from the 3 cameras, running object detection models from the EfficientDet family we evaluated in the previous section, with the



(a) *Network conditions and bandwidth allocations to each camera (service).*

(b) *Average accuracy across services, both predicted and actually completed.*

(c) *Trace of an object detector service where the propagation delay (RTT/2) dominates the response time.*

Figure 13: ***Production runs.*** *We deploy TURBO on a 5G-connected vehicle and measure its efficacy across three object detection services for the front, left, and right cameras. We find that TURBO effectively allocates the available bandwidth to its services (Figure 13a) to increase the overall accuracy averaged across all services (Figure 13b). The trace of a single service (Figure 13c) confirms that network latency is a key bottleneck and that our implementation adds minimal overhead.*

same 150ms SLO we use in simulation. Notably, we did not allow our control system to touch any car controls for safety reasons the car was always fully driven by the authors, with the control system responsible for collecting and processing input as the perception control stage.

Real-world utility from TURBO. Figure 13a shows a 30-second view of TURBO's allocations to each camera during a period of good network availability. During the wider 5-minute period surrounding this view, we saw 88% of requests successfully returned, enabling the front camera to upgrade its model 76% of the time, the left camera 86%, and the right camera 0%.[12] Across the three services, this translated to an absolute average increase of 4.1% pointers of accuracy. We consume a total of 105.6 MB of data over this window. Figure 13b shows TURBO's predicted average accuracy, vs the actual accuracy when factoring in missed SLOs causing fallback to on-car models.

Overall, we find our system successfully dynamically adapts to network conditions to make best use of network conditions, resulting in successful cloud offloads when network conditions allow. In times when cellular connectivity was poor, TURBO successfully maintained control pipeline integrity in the face of degraded network conditions, *always* proceeding with the control pipeline within SLO using local results when network connectivity was unreliable.

System microbenchmarks. We show a breakdown of the time spent in each component of our system (Section 5.6) from the perspective of a single service in Figure 13c. We find that our implementation provides minimal overhead and spends only 2.3 ms on serialization. In this trace, the propagation delay (RTT) of 58.9 ms is the largest source of latency, which may be caused by congestion on the 5G network and the large distance to our server. In contrast, the low transmission delay (5.6 ms upload, 0.4 ms download) is explained by the small size of the compressed image (19 KB).

We additionally profiled the runtime of our bandwidth allocation module, which we configured to run every 500ms as we updated our network measurements. We found the

[12] We note that under constrained bandwidth conditions, our method prioritized upgrading the left and front cameras over the right because the right camera sees fewer cars (due to driving on the right), so benefits less from better models. This is captured implicitly in the utility curves we derived offline.

mean runtime of our optimization formulation running on the on-vehicle compute to be 14ms, with a standard deviation of 1ms and a maximum of 23ms. Thus, the computational overhead of solving the ILP is minimal, rather the bottleneck is the frequency of measuring network conditions.

Network performance. In our run, we experienced worse network performance than expected: bandwidth conditions ranged from 0.80 Mbps (5th pct) to 55.86 Mbps (95th pct), and RTTs were high, starting at 60ms at the 5th pct and going up to hundreds of ms at the 95th pct. We suspect issues with the antenna for our setup, as literature reports considerably higher network performance: 5G deployments target a *minimum (5th-percentile)* user-experienced bandwidth of 50 Mbps for uplink and 100 Mbps for downlink [54], and prior work shows moving vehicle 5G network conditions to be expected at 80 Mbit/s (25th pct) to 160 Mbit/s (75th pct) for uplink bandwidth, and 12ms (25th pct) to 18ms (75th pct) for base station RTT ping latency [42, 100].

Cost. For our test-drive, we purchased an unlimited plan from a carrier with a monthly high-speed usage cap at 50GB. Our test run did not exceed this cap, however a continuous real-world deployment would need to purchase more data to run continuously throughout the month. In Appendix A.1, we estimate our total hourly cost of remote resources at $5.27, with $2.78 from the network (at the 10th global percentile) and $2.49 from compute for an H100. We emphasize that the true cost of cloud compute is likely lower due to better efficiency when operating at scale. Our method is tunable for cost-sensitive markets: operators can tune their utility curves to take into account cost-benefit, *e.g.,* by executing on cheaper GPUs and selectively using cloud models in challenging environments.

7 Discussion

Limitations. Our design does not consider scheduling the order of messages, *i.e.,* delaying transmission for one service to make more bandwidth available for another service. Likewise, we do not consider executing consecutive tasks in the cloud without returning back to the car which could further reduce data transmission. We focus on offloading parallelizable tasks, and leave scheduling extensions (*e.g.,* exploiting task dependencies) to future work. TURBO relies on 5G coverage for fast network bandwidths and low round-trip times. While 5G coverage is expanding, it is not available in every region [31]. Finally, TURBO requires the AV operator to configure f_s (Section 4.3) to ensure that the ILP maximizes the end-to-end performance of the AV system, as maximizing average accuracy across *all* services may result in maximal safety. We believe that configurations of f_s may be discovered using simulations and machine learning, and leave this area to future work.

Implications for AV design. Our results suggest that using cloud resources to run more accurate models can significantly improve accuracy, enabling AVs to make better-informed, high quality decisions which benefit safety. Beyond safety events, cloud models can reduce the frequency of remote interventions [123] by improving the AV's understanding of its surroundings. Fewer interventions improves the experience for riders, reduces the cost of operating AVs, and is an important metric tracked by government officials in assessing AV system safety [95].

At the same time, we note that adopting the cloud raises new concerns; the cellular connection is elevated to a more important role, which increases the need for security in the AV system as well as improvements to the performance, reliability, and availability of the cellular connection.

Implications for Mobile Network Stakeholders. If AVs integrate cloud computing, the

demand for cellular network resources will increase massively. In 2022, 283 million vehicles were registered in the United States [1] alone out of an estimated 1.5 billion vehicles in use worldwide [30]. If only a fraction of these vehicles use the cloud to enhance autonomous capabilities, mobile networks networks must support millions of new users with long-running, data-intensive streaming workloads. Moreover, AVs demand better uplink bandwidth and lower round-trip latencies. A key bottleneck to RTT is the hop from the 5G base station to the cellular core network, which accounts for most of the RTT according to [140]. Consequently, cellular network operators will need to prioritize performance improvements and potentially provide network guarantees for AVs and other safety-critical connected systems.

Applications beyond AVs. We believe that the benefits of using cloud hardware in real-time to boost accuracy has far-reaching applications. Beyond AVs, TURBO can benefit vehicles with limited autonomy, such as hands-free highway driving or self-parking. Similarly, autonomous drones use ML to process sensor data in real time. Drones exhibit even more stringent constraints on compute due to weight restrictions, making access to cloud resources an attractive option. Robotics [19], security systems, and vision-language models are other applications which benefit from real-time access to cloud resources to benefit accuracy.

8 Related Works

Autonomous driving with remote resources. Several works propose designs for decentralized systems which build on Vehicle-to-Vehicle (V2V) or Vehicle-to-Infrastructure (V2X) communication patterns to harness additional compute resources [78, 25, 118, 116], share data [67, 149, 98], or develop collaborative algorithms [20, 129, 91, 76]. While such approaches can improve the accuracy of AV services, prior works do not study accuracy-aware bandwidth allocation decisions under fluctuating network conditions. One prior work [106] shows that runtimes of AV-specific models in the cloud are sufficiently faster than on typical on-car compute to make room for cellular ping latencies, enabling net quicker car reaction times. However, this work does not account for bandwidth-induced delay, only measuring *ping* RTTs for the network delay. We show in Section 6.1 that naively including bandwidth-induced delay negates any cloud benefit.

Edge-cloud inference. How to partition a single neural network between mobile devices and datacenters is an active area of research [60, 99, 148]. While existing approaches reduce inference latency and energy consumption, they do not apply to real-time settings with multiple services and multiple potential model configurations. In the cloud, model serving systems [23, 104, 46] are designed to serve many different model configurations while meeting statistical SLOs, which is complementary to our work. Because these systems are deployed in datacenters, they do not consider bandwidth a key resource under contention which needs to be managed.

Reconfigurable video analytics systems. Several systems reconfigure the ML inference pipeline to react to changes in the input content [147, 141, 142, 57, 79] and to maximize real-time accuracy metrics [72, 108]. Video analytics systems like Reducto [73] and DDS [29] maintain high accuracy while reducing edge-cloud network traffic, but do not address contention for bandwidth. Ekya [11] and NoScope [59] use online learning to further improve the accuracy of video analytics. JCAB [128] proposes a similar joint optimization problem to allocate bandwidth across multiple video streams and maximize overall accuracy. In contrast to TURBO, JCAB targets real-time analytics instead of real-time safety-critical control. As such, JCAB targets a long-term average latency SLO, does not factor in RTT, and varies frame rates to trade-off accuracy and bandwidth instead of compression.

Bandwidth allocation algorithms. The problem of utility-aware bandwidth allocation has been studied in a variety of concrete settings [87, 77, 53], as well as more abstractly [35, 16]. Google's BwE paper [66] takes a similar approach to ours of decomposing overall utility into per-service utility curves, however they compose and optimize over their per-service curves differently as their objective is max-min fairness across flows. Furthermore, they perform their allocation centrally across a global WAN with different tasks and users. We examine the problem of bandwidth allocation from the perspective of the components components within the AV control application, which is part of the larger class of compound AI systems [107].

References

1 Federal Highway Administration. Table mv-1 - highway statistics 2022. https://www.fhwa.dot.gov/policyinformation/statistics/2022/mv1.cfm, November 2023.

2 National Highway Traffic Safety Administration. Automated Vehicles for Safety. https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety.

3 National Highway Traffic Safety Administration. Collision between vehicle controlled by developmental automated driving system and pedestrian. https://www.ntsb.gov/investigations/accidentreports/reports/har1903.pdf, mar 2018.

4 National Highway Traffic Safety Administration. Part 573 safety recall report 23e-086, November 2023.

5 Nefi Alarcon. Drive labs: How localization helps vehicles find their way. https://developer.nvidia.com/blog/drive-labs-how-localization-helps-vehicles-find-their-way/, January 2020.

6 Apollo. Apollo. https://github.com/ApolloAuto/apollo/. Accessed: 2024-9-5.

7 Argoverse. Argoverse. https://www.argoverse.org/.

8 Autoware. Autoware concepts. https://autowarefoundation.github.io/autoware-documentation/galactic/design/autoware-concepts/. Accessed 2024-9-18.

9 Baidu. Apollo 3.0 Software Architecture. https://tinyurl.com/mhd6dfka.

10 Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

11 Romil Bhardwaj, Zhengxu Xia, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, Nikolaos Karianakis, Kevin Hsieh, Paramvir Bahl, and Ion Stoica. Ekya: Continuous learning of video analytics models on edge compute servers. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 119–135, 2022.

12 Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. 2020. URL: https://arxiv.org/abs/2004.10934, arXiv:arXiv:2004.10934.

13 Eric Brandt. 2024 tesla model 3. https://www.kbb.com/tesla/model-3/, March 2024. Accessed 2024-6-11.

14 Bureau of Transportation Statistics. Average age of automobiles and trucks in operation in the united states, January 2024.

15 Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A Multimodal Dataset for Autonomous Driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, Seattle, WA, USA, June 2020. IEEE. doi:10.1109/CVPR42600.2020.01164.

16 Zhiruo Cao and E W Zegura. Utility max-min: an application-oriented bandwidth allocation scheme. In *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, volume 2, pages 793–801 vol.2. IEEE, 1999.

17 Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*. Springer, 2020.

18 Max Chafkin. Even after $100 billion, self-driving cars are going nowhere. https://www.bloomberg.com/news/features/2022-10-06/even-after-100-billion-self-driving-cars-are-going-nowhere, October 2022.

19 Kaiyuan Eric Chen, Yafei Liang, Nikhil Jha, Jeffrey Ichnowski, Michael Danielczuk, Joseph Gonzalez, John Kubiatowicz, and Ken Goldberg. Fogros: An adaptive framework for automating fog robotics deployment. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 2035–2042. IEEE, 2021.

20 Qi Chen, Xu Ma, Sihai Tang, Jingda Guo, Qing Yang, and Song Fu. F-cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3d point clouds. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pages 88–100, 2019.

21 Dah-Ming Chiu and Raj Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17(1):1–14, 1989. URL: https://www.sciencedirect.com/science/article/pii/0169755289900196, doi:https://doi.org/10.1016/0169-7552(89)90019-6.

22 Henry Claypool, Amitai Bin-Nun, and Jeffrey Gerlach. Self-Driving Cars: The Impact on People with Disabilities. *Newton, MA: Ruderman Family Foundation*, 2017.

23 Daniel Crankshaw, Xin Wang, Guilio Zhou, Michael J Franklin, Joseph E Gonzalez, and Ion Stoica. Clipper: A Low-Latency Online Prediction Serving System. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation (NSDI)*, pages 613–627, 2017.

24 Cruise. Cruise to launch robotaxi services in austin, phoenix before end of 2022. https://techcrunch.com/2022/09/12/cruise-to-launch-robotaxi-services-in-austin-phoenix-before-end-of-2022/.

25 Mingyue Cui, Shipeng Zhong, Boyang Li, Xu Chen, and Kai Huang. Offloading autonomous driving services via edge computing. *IEEE Internet of Things Journal*, 7(10):10535–10547, 2020.

26 Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

27 Derek Chiao, Johannes Deichmann, Kersten Heineke, Ani Kelkar, Martin Kellner, Elizabeth Scarinci, Dmitry Tolstinev. Autonomous vehicles moving forward: Perspectives from industry leaders. Technical report, McKinsey, January 2024.

28 Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

29 Kuntai Du, Ahsan Pervaiz, Xin Yuan, Aakanksha Chowdhery, Qizheng Zhang, Henry Hoffmann, and Junchen Jiang. Server-driven video streaming for deep learning inference. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 557–570, 2020.

30 Arndt Ellinghorst, Meike Becker, Neil Beveridge, Bob Brackett, Danielle Chigumira, Oswald Clint, Chad Dillard, Brian Foran, Venugopal Garre, Jay Huang, Cherry Leung, Mark Li, Zhihan Ma, A.M. (Toni) Sacconaghi, Jean Ann Salisbury, Deepa Venkateswaran, Lu Wang, Robert Wildhack, and Gunther Zechmann. Electric revolution 2021: From dream to scare to reality?, August 2021.

31 Ericsson. Network coverage forecast ericsson mobility report. https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/network-coverage, November 2020. Accessed: 2024-9-19.

32 ESnet. iperf3, 2024. URL: https://github.com/esnet/iperf.

33 Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9710–9719, 2021.

34 Scott Ettinger, Kratarth Goel, Avikalp Srivastava, and Rami Al-Rfou. Scaling motion forecasting models with ensemble distillation, 2024. arXiv:2404.03843.

35 A K Maulloo F P Kelly and D K H Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research*

Society, 49(3):237–252, 1998. arXiv:https://doi.org/10.1057/palgrave.jors.2600523, doi:10.1057/palgrave.jors.2600523.

36 Bill Fink and Rob Scott. nuttcp. URL: http://nuttcp.net/.

37 FiveThirtyEight. Uber pickups in new york city. https://www.kaggle.com/datasets/fivethirtyeight/uber-pickups-in-new-york-city. Accessed 2024-6-23.

38 John Forrest, Ted Ralphs, Stefan Vigerske, Haroldo Gambini Santos, John Forrest, Lou Hafer, Bjarni Kristjansson, jpfasano, EdwinStraver, Jan-Willem, Miles Lubin, rlougee, a andre, jp-goncal1, Samuel Brito, h-i gassmann, Cristina, Matthew Saltzman, tosttost, Bruno Pitrus, Fumiaki MATSUSHIMA, Patrick Vossler, Ron @ SWGY, and to st. coin-or/cbc: Release releases/2.10.12, August 2024. doi:10.5281/zenodo.13347261.

39 Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

40 Andrea Gemmet and Malea Martin. Driverless food delivery: Nuro teams up with uber eats to deploy autonomous vehicles in mountain view. https://www.paloaltoonline.com/news/2022/09/11/driverless-food-delivery-nuro-teams-up-with-uber-eats-to-deploy-autonomous-vehicles-in-mountain-view/, September 2022.

41 Moinak Ghoshal, Z Jonny Kong, Qiang Xu, Zixiao Lu, Shivang Aggarwal, Imran Khan, Yuanjie Li, Y Charlie Hu, and Dimitrios Koutsonikolas. An in-depth study of uplink performance of 5g mmwave networks. In *Proceedings of the ACM SIGCOMM Workshop on 5G and Beyond Network Measurements, Modeling, and Use Cases*, pages 29–35, 2022.

42 Moinak Ghoshal, Z. Jonny Kong, Qiang Xu, Zixiao Lu, Shivang Aggarwal, Imran Khan, Yuanjie Li, Y. Charlie Hu, and Dimitrios Koutsonikolas. An in-depth study of uplink performance of 5g mmwave networks. In *Proceedings of the ACM SIGCOMM Workshop on 5G and Beyond Network Measurements, Modeling, and Use Cases*, 5G-MeMU '22, page 2935, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3538394.3546042.

43 Ionel Gog, Sukrit Kalra, Peter Schafhalter, Joseph E Gonzalez, and Ion Stoica. D3: A Dynamic Deadline-Driven approach for Building Autonomous Vehicles. In *Proceedings of the Seventeenth European Conference on Computer Systems*, pages 453–471, 2022.

44 Ionel Gog, Sukrit Kalra, Peter Schafhalter, Matthew A. Wright, Joseph E. Gonzalez, and Ion Stoica. Pylot: A Modular Platform for Exploring Latency-Accuracy Tradeoffs in Autonomous Vehicles. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 8806–8813. IEEE, 2021.

45 Jacopo Guanetti, Yeojun Kim, and Francesco Borrelli. Control of connected and automated vehicles: State of the art and future challenges. *Annual Reviews in Control*, 45:18–40, January 2018. doi:10.1016/j.arcontrol.2018.04.011.

46 Arpan Gujarati, Reza Karimi, Safya Alzayat, Antoine Kaufmann, Ymir Vigfusson, and Jonathan Mace. Serving DNNs like Clockwork: Performance Predictability from the Bottom Up. In *Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, November 2020.

47 Florian Götz. The data deluge: What do we do with the data generated by avs? https://blogs.sw.siemens.com/polarion/the-data-deluge-what-do-we-do-with-the-data-generated-by-avs/, January 2021.

48 Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

49 Ralph Heredia. How to Cost-Effectively manage IoT data plans. https://www.zipitwireless.com/blog/how-to-cost-effectively-manage-iot-data-plans, June 2023. Accessed: 2024-6-15.

50 Honda. Multi-model brake master cylinder recall. *Honda News*, Jul 2023. URL: https://hondanews.com/en-US/honda-corporate/releases/release-13f25e90cfe47cd58453f1f710208486-multi-model-brake-master-cylinder-recall.

51 Dan Howdle. Worldwide mobile data pricing 2023. https://www.cable.co.uk/mobiles/worldwide-data-pricing/. Accessed: 2024-6-13.

52 Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17853–17862, 2023.

53 Xiaohong Huang, Tingting Yuan, and Maode Ma. Utility-optimized flow-level bandwidth allocation in hybrid sdns. *IEEE Access*, 6:20279–20290, 2018. doi:10.1109/ACCESS.2018.2820682.

54 International Telecommunications Union. Report ITU-R m.2410-0: Minimum requirements related to technical performance for IMT-2020 radio interface(s). Technical Report M.2410-0, International Telecommunications Union, 2017.

55 Jana Iyengar and Martin Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000, May 2021. URL: https://www.rfc-editor.org/info/rfc9000, doi:10.17487/RFC9000.

56 Satish Jeyachandran. Meet the 6th-generation waymo driver: Optimized for costs, designed to handle more weather, and coming to riders faster than before. https://waymo.com/blog/2024/08/meet-the-6th-generation-waymo-driver/, August 2024. Accessed: 2024-8-20.

57 Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. Chameleon: Scalable Adaptation of Video Analytics. In *Proceedings of the ACM Special Interest Group on Data Communication Conference (SIGCOMM)*, pages 253–266, 2018. URL: http://doi.acm.org/10.1145/3230543.3230574, doi:10.1145/3230543.3230574.

58 Gunnar Johansson and Kåre Rumar. Drivers' brake reaction times. *Human factors*, 13(1):23–27, 1971.

59 Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. Noscope: optimizing neural network queries over video at scale. volume 10, page 15861597. VLDB Endowment, aug 2017. doi:10.14778/3137628.3137664.

60 Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News*, 45(1):615–629, 2017.

61 Andrej Karpathy. [cvpr'21 wad] keynote - andrej karpathy, tesla. https://youtu.be/g6bOwQdCJrc, June 2021.

62 Christos Katrakazas, Mohammed Quddus, Wen-Hua Chen, and Lipika Deka. Real-time Motion Planning Methods for Autonomous On-Road Driving: State-of-the-art and Future Research Directions. *Transportation Research Part C: Emerging Technologies*, 60:416–442, 2015.

63 Kiwibot. Delivery robots for everyone! https://www.kiwibot.com/. Accessed: 2024-9-18.

64 Stepan Konev, Kirill Brodt, and Artsiom Sanakoyeu. Motioncnn: A strong baseline for motion prediction in autonomous driving. *arXiv preprint arXiv:2206.02163*, 2022.

65 KPMG. Self-driving cars: The next revolution. https://institutes.kpmg.us/content/dam/institutes/en/manufacturing/pdfs/2017/self-driving-cars-next-revolution-new.pdf.

66 Alok Kumar, Sushant Jain, Uday Naik, Anand Raghuraman, Nikhil Kasinadhuni, Enrique Cauich Zermeno, C Stephen Gunn, Jing Ai, Björn Carlin, Mihai Amarandei-Stavila, Mathieu Robin, Aspi Siganporia, Stephen Stuart, and Amin Vahdat. BwE: Flexible, hierarchical bandwidth allocation for WAN distributed computing. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 1–14, New York, NY, USA, August 2015. Association for Computing Machinery.

67 Swarun Kumar, Shyamnath Gollakota, and Dina Katabi. A cloud-assisted design for autonomous driving. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 41–46, 2012.

68 Marie Labrie. Volvo cars, zoox, saic and more join growing range of autonomous vehicle makers using new nvidia drive solutions. https://nvidianews.nvidia.com/news/volvo-cars-

zoox-saic-and-more-join-growing-range-of-autonomous-vehicle-makers-using-new-nvidia-drive-solutions, April 2021.

69 Lambda. Gpu cloud. https://lambdalabs.com/service/gpu-cloud. Accessed 2024-6-23.

70 Zhaoqi Leng, Pei Sun, Tong He, Dragomir Anguelov, and Mingxing Tan. Pvtransformer: Point-to-voxel transformer for scalable 3d object detection, 2024. arXiv:2405.02811.

71 Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, et al. Yolov6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*, 2022.

72 Mengtian Li, Yuxiong Wang, and Deva Ramanan. Towards Streaming Perception. In *Proceedings of the European Conference on Computer Vision (ECCV)*, August 2020.

73 Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi Netravali. Reducto: On-camera filtering for resource-efficient real-time video analytics. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 359–376, 2020.

74 Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach, Md E. Haque, Lingjia Tang, and Jason Mars. The Architectural Implications of Autonomous Driving: Constraints and Acceleration. In *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 751–766, 2018. URL: http://doi.acm.org/10.1145/3173162.3173191, doi:10.1145/3173162.3173191.

75 Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft CoCo: Common Objects in Context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014.

76 Bing Liu, Qing Shi, Zhuoyue Song, and Abdelkader El Kamel. Trajectory planning for autonomous intersection management of connected vehicles. *Simulation Modelling Practice and Theory*, 90:16–30, 2019.

77 Changbin Liu, Lei Shi, and Bin Liu. Utility-based bandwidth allocation for triple-play services. In *Fourth European Conference on Universal Multiservice Networks (ECUMN'07)*, pages 327–336, 2007. doi:10.1109/ECUMN.2007.58.

78 Luyang Liu, Hongyu Li, and Marco Gruteser. Edge assisted real-time object detection for mobile augmented reality. In *The 25th annual international conference on mobile computing and networking*, pages 1–16, 2019.

79 Shaoshan Liu, Liangkai Liu, Jie Tang, Bo Yu, Yifan Wang, and Weisong Shi. Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, 107(8):1697–1716, 2019.

80 Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

81 Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022.

82 Yiren Lu, Justin Fu, George Tucker, Xinlei Pan, Eli Bronstein, Rebecca Roelofs, Benjamin Sapp, Brandyn White, Aleksandra Faust, Shimon Whiteson, et al. Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7553–7560. IEEE, 2023.

83 Cade Metz, Jason Henry, Ben Laffin, Rebecca Lieberman, and Yiwen Lu. How self-driving cars get help from humans hundreds of miles away. *The New York Times*, September 2024.

84 Michele Bertoncello, and Dominik Wee. Ten Ways Autonomous Driving Could Redefine the Automotive World. https://tinyurl.com/2srpyv8d.

85 Microsoft. What is quantum computing. https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-quantum-computing. Accessed: 2024-9-19.

86 Norman Mu, Jingwei Ji, Zhenpei Yang, Nate Harada, Haotian Tang, Kan Chen, Charles R. Qi, Runzhou Ge, Kratarth Goel, Zoey Yang, Scott Ettinger, Rami Al-Rfou, Dragomir Anguelov, and Yin Zhou. Most: Multi-modality scene tokenization for motion prediction, 2024. arXiv:2404.19531.

87 Kanthi Nagaraj, Dinesh Bharadia, Hongzi Mao, Sandeep Chinchali, Mohammad Alizadeh, and Sachin Katti. Numfabric: Fast and flexible bandwidth allocation in datacenters. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, page 188201, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2934872.2934890.

88 Arvind Narayanan, Xumiao Zhang, Ruiyang Zhu, Ahmad Hassan, Shuowei Jin, Xiao Zhu, Xiaoxuan Zhang, Denis Rybkin, Zhengxuan Yang, Zhuoqing Morley Mao, Feng Qian, and Zhi-Li Zhang. A variegated look at 5G in the wild: performance, power, and QoE implications. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, pages 610–625, New York, NY, USA, August 2021. Association for Computing Machinery.

89 National Highway Traffic Safety Administration. NHTSA 2022 annual report safety recalls. Technical report, National Highway Traffic Safety Administration, March 2023.

90 Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple & efficient attention networks. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2980–2987. IEEE, 2023.

91 Hieu Ngo, Hua Fang, and Honggang Wang. Cooperative perception with v2v communication for autonomous vehicles. *IEEE Transactions on Vehicular Technology*, 72(9):11122–11131, 2023.

92 NVIDIA. Drive agx orin developer kit. https://developer.nvidia.com/drive/agx. Accessed 2024-6-11.

93 NVIDIA. Nvidia drive developer faq. https://developer.nvidia.com/drive/faq. Accessed 2024-6-23.

94 NVIDIA. Nvidia h100 tensor core gpu. https://www.nvidia.com/en-us/data-center/h100/. Accessed 2024-6-11.

95 State of California Department of Motor Vehicles. Disengagement reports. https://www.dmv.ca.gov/portal/vehicle-industry-services/autonomous-vehicles/disengagement-reports/.

96 Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 2016.

97 Rafael Padilla and Amy Roberts. Hugging face object detection leaderboard. https://huggingface.co/blog/object-detection-leaderboard, September 2023. Accessed: 2024-8-22.

98 Hang Qiu, Fawad Ahmad, Fan Bai, Marco Gruteser, and Ramesh Govindan. Avr: Augmented vehicular reality. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 81–95, 2018.

99 Moo-Ryong Ra, Anmol Sheth, Lily Mummert, Padmanabhan Pillai, David Wetherall, and Ramesh Govindan. Odessa: enabling interactive perception applications on mobile devices. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 43–56, 2011.

100 Mohammad Razzaghpour, Carsten Bockelmann, Armin Dekorsy, Jan Hensel, Wilhelm Jochim, and Mehmet Kus. Empirical evaluation of bit rate and latency in a private 5g cell for slow-speed vehicles in an urban environment. In *2024 IEEE Globecom Workshops (GC Wkshps)*, pages 1–7, 2024. doi:10.1109/GCWkshp64532.2024.11100938.

101 Brian Tefft Rebecca Steinbach. American driving survey: 2022. Technical report, AAA Foundation for Traffic Safety, September 2023.

102 Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Proceedings of the International Conferences on Advances in Neural Information Processing Systems (NeurIPS)*, pages 91–99, 2015.

103 Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2P2: A Reparameterized Pushforward Policy for Diverse, Precise Generative Path Forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 772–788, 2018.

104 Francisco Romero, Qian Li, Neeraja J Yadwadkar, and Christos Kozyrakis. Infaas: Automated model-less inference serving. In *USENIX Annual Technical Conference*, pages 397–411, 2021.

105 J.S. Roy, Stuart A. Mitchell, Christophe-Marie Duquesne, Franco Peschiera, and Phillips Antony. Pulp. https://github.com/coin-or/pulp. Accessed 2024-9-14.

106 Peter Schafhalter, Sukrit Kalra, Le Xu, Joseph E Gonzalez, and Ion Stoica. Leveraging cloud computing to make autonomous vehicles safer. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5559–5566. IEEE, 2023.

107 Daniel Seita and Matei Zaharia, Omar Khattab, Lingjiao Chen, Jared Quincy Davis, Heather Miller, Chris Potts, James Zou, Michael Carbin, Jonathan Frankle, Naveen Rao, Ali Ghodsi. The shift from models to compound AI systems. https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/. Accessed: 2024-2-20.

108 Gur-Eyal Sela, Ionel Gog, Justin Wong, Kumar Krishna Agrawal, Xiangxi Mo, Sukrit Kalra, Peter Schafhalter, Eric Leong, Xin Wang, Bharathan Balaji, Joseph E Gonzalez, and Ion Stoica. Context-aware streaming perception in dynamic environments. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

109 Amazon Web Services. aws/s2n-quic. original-date: 2020-06-25T18:27:25Z. URL: https://github.com/aws/s2n-quic.

110 Jaime Sevilla, Lennart Heim, Anson Ho, Tamay Besiroglu, Marius Hobbhahn, and Pablo Villalobos. Compute trends across three eras of machine learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.

111 Aditya Sharma. Mean average precision (mAP) using the COCO evaluator. https://pyimagesearch.com/2022/05/02/mean-average-precision-map-using-the-coco-evaluator/, May 2022. Accessed: 2024-6-25.

112 Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. *Advances in Neural Information Processing Systems*, 35:6531–6543, 2022.

113 Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Mtr++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

114 Anton Shilov. Intel's gaudi 3 will cost half the price of nvidia's h100. https://www.tomshardware.com/pc-components/cpus/intels-gaudi-3-will-cost-half-the-price-of-nvidias-h100, June 2024.

115 Anton Shilov. Nvidia's h100 ai gpus cost up to four times more than amd's competing mi300x amd's chips cost $10 to $15k apiece; nvidia's h100 has peaked beyond $40,000: Report. https://www.tomshardware.com/tech-industry/artificial-intelligence/nvidias-h100-ai-gpus-cost-up-to-four-times-more-than-amds-competing-mi300x-amds-chips-cost-dollar10-to-dollar15k-apiece-nvidias-h100-has-peaked-beyond-dollar40000, February 2024.

116 Fei Sun, Fen Hou, Nan Cheng, Miao Wang, Haibo Zhou, Lin Gui, and Xuemin Shen. Cooperative task scheduling for computation offloading in vehicular cloud. *IEEE Transactions on Vehicular Technology*, 67(11):11049–11061, 2018.

117 Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Sheng Zhao, Shuyang Cheng, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and

Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. 2019. URL: https://arxiv.org/abs/1912.04838, arXiv:arXiv:1912.04838.

118 Yuxuan Sun, Xueying Guo, Sheng Zhou, Zhiyuan Jiang, Xin Liu, and Zhisheng Niu. Learning-based task offloading for vehicular cloud computing systems. In *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018.

119 Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019. URL: https://proceedings.mlr.press/v97/tan19a.html.

120 Mingxing Tan, Ruoming Pang, and Quoc V. Le. EfficientDet: Scalable and Efficient Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

121 Jigang Tang, Songbin Li, and Peng Liu. A review of lane detection methods based on deep learning. *Pattern Recognition*, 111:107623, 2021. URL: https://www.sciencedirect.com/science/article/pii/S003132032030426X, doi:https://doi.org/10.1016/j.patcog.2020.107623.

122 The Waymo Team. Waymo significantly outperforms comparable human benchmarks over 7+ million miles of rider-only driving. https://waymo.com/blog/2023/12/waymo-significantly-outperforms-comparable-human-benchmarks-over-7-million/, December 2023.

123 The Waymo Team. Fleet response: Lending a helpful hand to waymos autonomously driven vehicles. https://waymo.com/blog/2024/05/fleet-response/, May 2024.

124 Siyu Teng, Xuemin Hu, Peng Deng, Bai Li, Yuchen Li, Yunfeng Ai, Dongsheng Yang, Lingxi Li, Zhe Xuanyuan, Fenghua Zhu, et al. Motion planning for autonomous driving: The state of the art and future perspectives. *IEEE Transactions on Intelligent Vehicles*, 2023.

125 Tesla. Robotaxi. https://www.tesla.com/robotaxi. Accessed: 2025-9-29.

126 Nicolo Valigi. Lessons Learned Building a Self-Driving Car on ROS. https://roscon.ros.org/2018/presentations/ROSCon2018_LessonsLearnedSelfDriving.pdf, 2018.

127 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

128 Can Wang, Sheng Zhang, Yu Chen, Zhuzhong Qian, Jie Wu, and Mingjun Xiao. Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 257–266, 2020. doi:10.1109/INFOCOM41043.2020.9155524.

129 Nannan Wang, Xi Wang, Paparao Palacharla, and Tadashi Ikeuchi. Cooperative autonomous driving for traffic congestion avoidance through vehicle-to-vehicle communications. In *2017 IEEE Vehicular Networking Conference (VNC)*, pages 327–330. IEEE, 2017.

130 Wei-Hua Wang, Marimuthu Palaniswami, and Steven H. Low. Application-oriented flow control: fundamentals, algorithms and fairness. *IEEE/ACM Trans. Netw.*, 14(6):12821291, dec 2006. doi:10.1109/TNET.2006.886318.

131 Waymo. Introducing the 5th Generation Waymo Driver. https://blog.waymo.com/2020/03/introducing-5th-generation-waymo-driver.html.

132 Waymo. Motion prediction. https://waymo.com/open/challenges/2024/motion-prediction/. Accessed: 2024-8-20.

133 Waymo. Next stop for waymo one: Los angeles. https://waymo.com/blog/2022/10/next-stop-for-waymo-one-los-angeles.html.

134 Waymo. Waymo safety impact. https://waymo.com/safety/impact/. Accessed 2024-9-18.

135 Waymo. Waymo Safety Report: On the Road to Fully Self-Driving. https://storage.googleapis.com/sdc-prod/v1/safety-report/SafetyReport2018.pdf.

136 Waymo. Scaling waymo one safely across four cities this year. https://waymo.com/blog/2024/03/scaling-waymo-one-safely-across-four-cities-this-year/, March 2024.

137 Ross Wightman. efficientdet-pytorch. https://github.com/rwightman/efficientdet-pytorch. Accessed: 2024-9-6.

138 Benjamin Wolfe, Bobbie Seppelt, Bruce Mehler, Bryan Reimer, and Ruth Rosenholtz. Rapid holistic perception and evasion of road hazards. *Journal of experimental psychology: general*, 149(3):490, 2020.

139 Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger. *arXiv preprint arXiv:2312.10035*, 2023.

140 Dongzhu Xu, Anfu Zhou, Xinyu Zhang, Guixian Wang, Xi Liu, Congkai An, Yiming Shi, Liang Liu, and Huadong Ma. Understanding operational 5g: A first measurement study on its coverage, performance and energy consumption. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 479–494, 2020.

141 Ran Xu, Jayoung Lee, Pengcheng Wang, Saurabh Bagchi, Yin Li, and Somali Chaterji. Litereconfig: Cost and content aware reconfiguration of video object detection systems for mobile gpus. In *Proceedings of the Seventeenth European Conference on Computer Systems*, pages 334–351, 2022.

142 Ran Xu, Fangzhou Mu, Jayoung Lee, Preeti Mukherjee, Somali Chaterji, Saurabh Bagchi, and Yin Li. Smartadapt: Multi-branch object detection framework for videos on mobiles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2528–2538, 2022.

143 Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

144 Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2636–2645, 2020.

145 Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022.

146 Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel Ni, and Harry Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. In *International Conference on Learning Representations*, 2022.

147 Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J Freedman. Live Video Analytics at Scale with Approximation and Delay-Tolerance. In *Proceedings of the 14^{th} USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2017.

148 Shigeng Zhang, Yinggang Li, Xuan Liu, Song Guo, Weiping Wang, Jianxin Wang, Bo Ding, and Di Wu. Towards real-time cooperative deep inference over the cloud and edge end devices. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(2):1–24, 2020.

149 Xumiao Zhang, Anlan Zhang, Jiachen Sun, Xiao Zhu, Y Ethan Guo, Feng Qian, and Z Morley Mao. Emp: Edge-assisted multi-vehicle perception. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 545–558, 2021.

150 Zhuofan Zong, Guanglu Song, and Yu Liu. Detrs with collaborative hybrid assignments training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6748–6758, 2023.

151 Zoox. Zoox. https://zoox.com/. Accessed: 2024-9-18.

A Supplementary Material

A.1 Economic Feasibility

In this section, we analyze whether our approach is feasible when taking into account the cost of network transmission (Appendix A.1.1) and compute (Appendix A.1.2).

A.1.1 Network Cost

Commercial cellular network usage is charged primarily by the GB [49]. We conduct an analysis of consumer-marketed cellular data plans reported in the Cable.co.uk global mobile data pricing dataset [51]. Table 2 shows the cheapest *consumer*-facing (*i.e.,* SIM card) cost per GB of data in a selection of countries, along with the computed cost per hour of streaming an average of 100 Mbps of data continuously. We note that we expect wholesale pricing, especially geofenced to a particular region, to be considerably cheaper.

We see that prices vary widely from as low as \$0.001/GB in Israel, to \$0.75/GB in the US, up to over \$2 in Norway. This wide range in pricing requires careful consideration in deployment: in countries such as Israel, the price of cellular data transmission running our method is trivial at \$0.04 per hour of driving, assuming an average utilization of 100 Mbps. In some other countries, including the U.S. with a price of \$33.76 per hour, prices are considerably higher and present an economic obstacle at present to using remote resources. However, we note that at or below the 10th percentile of global prices which includes major markets such as India, Italy, and China mobile networks are cost-effective at \$2.78 per hour of driving. We expect much of the rest of the world to follow to these prices, as median price per GB has continuously decreased 4× over the years our dataset covers, 2019-2024, from \$5.25 to \$1.28.

In the short-term, in countries with high cellular data prices, operators may choose to reduce costs by selectively utilizing remote resources to aid in high-stress driving environments *e.g.,* during poor visibility due to weather and busy urban areas. Alternatively, it would be feasible for AV fleet operators to deploy their own dedicated locale-specific wireless network at cheaper cost.

A.1.2 Compute Cost

Cloud providers offer competitive access to GPUs: Lambda Labs hourly pricing ranges from \$0.80 for an NVIDIA A6000 GPU to \$2.49 for an NVIDIA H100 GPU [69].

Cloud compute offers a number of additional valuable advantages not available on the car. Cloud access allows operators to configure which compute resources to use based on compute requirements and cost sensitivity. Remote resources cannot be stolen or damaged in an accident. AV fleet operators can take advantage of statistical multiplexing to share a smaller set of compute for their fleet Appendix A.1.3. Furthermore, model serving systems can optimize resource utilization by batching and scheduling requests [23, 104, 46], resulting in further price improvements.

Total cost. We estimate total hourly cost of remote resources at \$5.27, with \$2.78 from the network (at the 10th global percentile) and \$2.49 from compute for an H100. We emphasize that the true cost of cloud compute is likely lower due to better efficiency when operating at scale.

Rank	Country	$/GB	$/Hour
1	Singapore	$0.07	$3.30
2	Netherlands	$0.36	$16.08
3	Norway	$2.09	$94.14
4	United States	$0.75	$33.76
5	Finland	$0.26	$11.62
–	China	$0.27	$12.28
–	Israel	$0.001	$0.04
–	*10th pct*	$0.062	$2.78
–	*Median*	$0.37	$16.84

Table 2: *Network costs ranked highest by AV readiness score [65]. We include China as a major AV market [27], Israel as the cheapest cellular market, and the 10th percentile and median global country by network price. Hourly rates assume an average constant network utilization of 100 Mbps.*

A.1.3 Cloud Compute Multiplexing

Sharing upgraded compute costs across a fleet of vehicles presents a significant opportunity to further reduce costs over upgrading on-vehicle compute due to statistical multiplexing of cloud resources. Though cars still need to retain their own GPUs, shared upgrades to cloud resources can instantly benefit fleets of AVs. The average driver in the U.S. drives only 60.2 minutes per day [101], *i.e.,* a vehicle utilization of 4.2%. This under-utilization is more pronounced for personal vehicles than autonomous ride-hailing services, which we estimate to be ∼59% based on the ratio of peak to average hourly Uber rides in New York City [37]. Considering this under-utilization, the cost of purchasing a single H100 GPU (∼$40k [115]) is equivalent to renting an H100 in the cloud for an 44 years for the average American driver, and 3 years for the average autonomous ride-hailing vehicle.

A.2 Practical Necessity of TURBO

We acknowledge that there are limited deployments of commercial AVs today [133, 24, 136] which outperform humans on safety benchmarks [134, 135], and rely on this capability to provide a fallback when network connectivity is unavailable. However, "safety" as both a concept and a metric is continuous, measured as a rate of incident occurrence [95, 134, 135], and AVs must merely exceed human-level safety for deployment [122]. TURBO seeks to improve safety beyond what on-car systems can provide today and presents an opt-in solution for AV providers to improve the accuracy of their services by leverage cloud and network infrastructure.

For AVs with different sets of economic and technical requirements (*e.g.,* operating in low-stakes environments and with little compute, such as autonomous delivery robots [63, 151, 40]), TURBO may provide a viable method of improving existing or expanding functionality such as safe high-speed operation or decreasing the rate of human intervention [123].

As AVs deployments expand, we expect to see more "outdated" AV models on the road[13]. As SOTA compute hardware performance, and correspondingly SOTA model sizes and requirements, continues to rapidly increase [110, 81, 28], outdated hardware will prevent older vehicles from utilizing the latest model advancements. While upgrades to on-vehicle compute hardware are possible, they may be difficult to roll out in practice. As a reference, fix

[13] The average lightweight vehicle age in the U.S. is 12.5 years [14].

rates for recalls are only 52-64% [89] despite being free, mandatory upgrades that mitigate critical safety risks (*e.g.,* faulty brake systems [50]). TURBO provides access to SOTA cloud hardware, enabling older AVs to use highly accurate, SOTA models.

Finally, SOTA hardware requires increasingly stringent operating environments, ranging from high power and cooling needs for GPUs [74], to strong intolerance to movement or temperature changes for quantum computing [85]. As a result, the only way to integrate methods that require such hardware *is* via the network, and TURBO presents a system that can manage this integration. Hence, we firmly believe that such a system is useful today and will become more applicable to the real world as current technical and economic trends continue.